A Framework for Leveraging Partially-Labeled Data for Product Attribute-Value Identification

D. Subhalingam*
KnowDis AI
Delhi, India
subhalingam.d@knowdis.ai

Keshav Kolluru*
KnowDis AI
Delhi, India
keshav.kolluru@knowdis.ai

Mausam
Indian Institute of Technology Delhi
Delhi, India
mausam@cse.iitd.ac.in

Saurabh Singal
KnowDis AI
Delhi, India
saurabh@knowdis.ai

Abstract

In the e-commerce domain, the accurate extraction of attribute-value pairs (e.g., Brand: *Apple*) from product titles and user search queries is crucial for enhancing search and recommendation systems. A major challenge with neural models for this task is the lack of high-quality training data, as the annotations for attribute-value pairs in the available datasets are often incomplete. To address this, we introduce GenToC, a model designed for training directly with partially-labeled data, eliminating the necessity for a fully annotated dataset. GenToC employs a marker-augmented generative model to identify potential attributes, followed by a token classification model that determines the associated values for each attribute. GenToC outperforms existing state-of-the-art models, exhibiting upto 56.3% increase in the number of accurate extractions. Furthermore, we utilize GenToC to regenerate the training dataset to expand attribute-value annotations. This bootstrapping substantially improves the data quality for training other standard NER models, which are typically faster but less capable in handling partially-labeled data, enabling them to achieve comparable performance to GenToC. Our results demonstrate GenToC's unique ability to learn from a limited set of partially-labeled data and improve the training of more efficient models, advancing the automated extraction of attribute-value pairs. Finally, our model has been successfully integrated into IndiaMART, India's largest B2B e-commerce platform, achieving a significant increase of 20.2% in the number of correctly identified attribute-value pairs over the existing deployed system while achieving a high precision of 89.5%. We have released the code for GenToC model at https://github.com/KnowDisAI/GenToC.

*Both authors contributed equally to this research.

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709427

CCS Concepts

• Information systems → Information retrieval query processing; **• Computing methodologies → Information extraction**; **• Applied computing →** *Online shopping*.

Keywords

Attribute-Value Extraction; E-commerce Search; Partially-labeled Data; Recommendation Systems

ACM Reference Format:
D. Subhalingam, Keshav Kolluru, Mausam, and Saurabh Singal. 2025. A Framework for Leveraging Partially-Labeled Data for Product Attribute-Value Identification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709427

1 Introduction

The rapid expansion of e-commerce has led to a significant increase in the variety and complexity of products available online. Each product typically includes a set of attributes such as Brand, Model Name, Color, with distinct values like *Boat*, *Rockerz 255 Pro*, *Raging Red* (as demonstrated in Table 1). These attributes and values help consumers locate and select their desired products. Automatic attribute-value identification is a well-studied problem in the e-commerce literature [1, 13, 14, 17, 20, 25, 27, 30]. It has been investigated in various contexts that involve additional metadata, such as product descriptions [22, 28], knowledge graphs [16] or images [9, 31]. In this study, we focus on automated extraction using only textual information [27, 30], such as product titles and user search queries.

To illustrate the practical utility of attribute-value extraction systems, we examine their usage in IndiaMART[1], India's largest B2B e-commerce platform, where our system is currently deployed. In this platform, our attribute-value extraction system serves a dual purpose, enhancing both product listings and user searches. The attribute-value pairs provided in product listings by sellers are often incomplete and can be enhanced using extractions from the product title. In addition, the system is used to extract attribute-value pairs from user search queries, and these extracted pairs are matched with attribute-value pairs from the retrieved product listings to highlight the matching ones. This *dynamic feature highlighting* of

[1]https://www.indiamart.com

Table 1: Complete collection of attribute-value pairs covering all words in the specified product title, with only the attributes Brand, Model Name and Color (marked with *) are included in the training data and the remaining attributes are extracted by our GenToC model.

Product Title: *Boat Rockerz 255 Pro Raging Red Bluetooth Neckband*

Attribute	Value
Brand*	*Boat*
Model Name*	*Rockerz 255 Pro*
Color*	*Raging Red*
Connectivity	*Bluetooth*
Headphone Type	*Neckband*

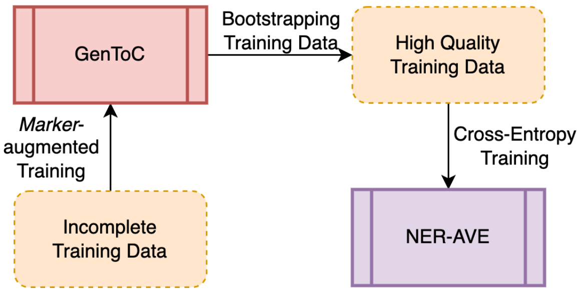

Figure 1: Overall framework. We train GenToC system with markers to effectively learn from incomplete training data. It is then used to bootstrap high-quality training data to train the real-time NER attribute-value extraction (AVE) system.

product attribute-value pairs based on the search query enables users to easily find the products that meet their requirements. As an illustration of dynamic feature highlighting, when a user searches for *"10 tier shoe rack"*, the system detects the attribute-value pair Number of shelves: *10*. This information is then used to highlight the corresponding feature Available number of shelves: *10* available for a listing with the product title *"Steel Shoe Rack"*, in the search results. This leads to an enriched user search experience.

The predominant approach for acquiring training datasets for attribute-value extraction tasks typically relies on leveraging product listings and associated attribute-value pairs as furnished by vendors on e-commerce platforms. This method is directly dependent on the comprehensiveness and accuracy of the data that vendors provide. However, it is common practice for vendors to list these details in an incomplete or inconsistent manner. Such gaps and irregularities in the data present a substantial obstacle, as the development of effective deep learning models for attribute-value extraction is contingent upon the availability of large-scale, high-quality training data. Given that attribute-value extraction systems are also employed on user query traffic to accurately capture user search intentions [16], it is essential to develop light-weight systems that provide real-time responses while managing the substantial workloads prevalent in e-commerce platforms.

To address these dual challenges of incomplete training data and the need for real-time responses, we propose a novel framework. First, we use a specialized model designed to learn from the incomplete data. This model is then used to regenerate the training data, thereby enabling the training of a faster model that can deliver real-time responses. This overall framework is illustrated in Figure 1.

To address the challenges of incomplete training data, we introduce GenToC, a novel model that uses a specialized *marker* learning strategy. GenToC is a two-stage neural model that decomposes the task into two distinct phases. First, it identifies all attributes within the input text, which can be either a product title or a user search query. Then, it extracts the corresponding values for each attribute. Specifically, GenToC employs a **Gen**erative Seq2Seq model in the initial stage and a **To**ken **C**lassification model in the subsequent stage to accurately extract attribute-value pairs from the input text. The first-stage generative model is designed to output all relevant attributes concatenated with each other using a delimiter. The second-stage token classification model receives each attribute name along with the original input query and determines for each word in the text whether it belongs to the value of the input attribute. For example, for the input text in Table 1, the first stage model can identify attributes such as Brand, Model Name, and Color. Then, the second stage model assigns *Boat*, *Rockerz 255 Pro* and *Raging Red* as the values for the corresponding attributes.

To foster the model's learning capacity using partially-labeled data, we incorporate special *markers* during the training phase of the first-stage generative model. These markers are used to highlight the words in the input query that are present as attribute values in the training data. For example, in Table 1, markers will be added to the words *Boat*, *Rockerz*, *255*, *Pro*, *Raging*, *Red* (at token level) because these are identified as attribute values in the training data. Through the strategic use of these markers, the model learns to form associations between attributes and highlighted words, that every highlighted word is a potential source of attribute information. This helps the model during inference, where we apply a marker to *every* token in the input. Applying a marker to each token serves as a signal to the model to regard every word as a possible source of attribute information, since similar correlations were observed during training. Thus, the model tends to produce a wider range of attributes and is not limited to mimicking the incomplete patterns observed in training. Existing conventional models lack this generalization capability, making it difficult to utilize partially labeled data effectively.

The second-stage token classification model accepts both the attribute name and the original query as input. It is tasked with classifying each word within the query to ascertain whether that signifies a value for the given input attribute. In the training phase, we use the attributes that are already known, whereas during inference, we rely on the attributes predicted by the first-stage model. Due to the cascading design of GenToC, errors from the first-stage model can potentially escalate, adversely affecting the second-stage model's performance. To fortify the second-stage model against such compounded inaccuracies, we implement a 'Value Pruning' method. This technique equips the value-extraction model with the ability to generate null outputs for any incorrect attributes that might be output by the first-stage attribute-extraction model.

Compared to existing state-of-the-art NER models [30] and Generative models [22] used for attribute-value extraction, GenToC achieves an increase of 16.2% and 18.2% in F1-score, respectively, on the test set. Despite these substantial performance gains, its speed of 90 ms per query presents a limitation for deployment.

Motivated by the objective of creating a faster model, we use GenToC to predict attribute-value pairs for each query in the training data and regenerate the dataset. This bootstrapping process improves attribute-value tagging in the training dataset. By training an NER model using this augmented dataset, we achieve an increase of 16.8% in F1-score compared to using the original incomplete dataset. It is also now comparable to GenToC in terms of F1-score, while maintaining an inference speed of under 9 ms per query. The GenToC-bootstrapped NER system is currently deployed on IndiaMART, replacing a rule-based system and has already served over 200M requests. Compared to the previous system, our new system has improved the recall of attribute-value extraction by over 20% on product titles and has resulted in a 9% increase in user search queries that activate dynamic feature highlighting.

To summarize, the major contributions of our work include

- Introducing GenToC, a novel two-stage architecture featuring a **Gen**erative Seq2Seq model for attribute extraction followed by a **To**ken **C**lassification model for value mapping, scalable to tens of thousands of attributes.
- Utilizing *markers*-based learning in GenToC's first stage to handle incomplete attribute-value tagging in training data.
- Surpassing existing methods with a 16.2% F1-score increase.
- Enhancing the training data using bootstrapping, thereby boosting the F1-score of NER model trained with it by 16.8%.
- Deploying the system in a leading B2B platform, where it has served over 200M requests in production so far.

2 Related Work

Attribute-value extraction [1, 12–14, 17, 20, 25, 27, 30] has been a significant topic of study in the realm of e-commerce, with a wealth of research targeting the extraction of product details from various modalities, including purely text-based methods [25, 27] as well as those that incorporate images [9, 24, 31]. A common limitation of these methods is their assumption of high-quality training data, which fails in many real-world scenarios where the training data is created using distant supervision and hence potentially incomplete [23, 27]. Some publicly available datasets like MAVE [28] offer high-quality data (achieving over 98% F1-score), but our work specifically addresses the more challenging case of partially-labeled settings, where existing models struggle.

Few works, such as Zhang et al. [29], Zheng et al. [30] focus on improving the training data quality in attribute-value extraction tasks but are severely limited to operating on a small number of attributes only. For instance, Zhang et al. [29] rely on a subset of strongly-labeled data to train a teacher network which in turn creates training data for a student network. The requirement of having a strongly-labeled subset limits them to operate on 13 attributes. Zheng et al. [30] also relies on a small set of labeled instances to be used in an active learning setting to collect good examples for manual annotation. They apply the technique to only one attribute per dataset. In contrast, GenToC introduces a fundamentally new model design that can handle incomplete training data without requiring any completely labeled subsets. This allows the model to scale to tens of thousands of attributes.

In terms of modelling, some of the earliest works on attribute-value extraction primarily employed rule-based extraction methods. They utilized a specialized seed dictionary or vocabulary to identify key phrases and attributes [6, 7, 26]. Moving beyond rule-based systems, several neural models have also been proposed for this task in the recent past [17]. Neural architectures for this task can be broadly divided into two categories – token classification [23, 27, 28] or generative [18, 22] models. Token classification models utilize NER models to identify the spans in the input text corresponding to an attribute. On the other hand, generative models utilize Seq2Seq models to produce relevant attribute-value pairs from a specified input. Our GenToC model makes clever use of both types of architectures, using a generative model for attribute extraction and a token-classification model for value extraction. We compare extensively with both types of architectures and show significant gains achieved by GenToC.

Token classification systems that rely on NER use atomic embeddings for encoding attributes, which makes it challenging to handle long-tail and complex attributes with less training data. Other token-classification systems utilize a BERT+LSTM model to embed each attribute separately [27]. The embedded representation attends over the input query and identifies the corresponding value. However, this approach encounters scalability issues as the model must check for values using every possible attribute. Some generative models [9] are limited in their operation to a relatively small set of attributes (in their case, 38). This is because they use a Seq2Seq model pass for every attribute to identify the relevant values. Other generative models [22] are applicable to larger attribute sets as they generate the attribute name as well. Similarly, GenToC can handle large attribute sets due to a dedicated module that generates all the relevant attributes. Like other generative models, GenToC employs compositional encoding for attributes, which allows it to capture the semantic meaning of attribute names by considering the words they contain. So it is better at handling long-tail and complex attributes compared to NER models. But the generative nature of the model comes with a higher response time. We summarize the characteristics of the different models in Table 2.

Recently, LLMs have also been explored as possible solutions for the attribute-value extraction task. [2] explores the use of GPT-3.5/4 and Llama-3-70B in zero-shot and few-shot scenarios. However, it also requires attribute names as input to the model and cannot scale to 25K attributes in our dataset. [4] proposes to ensemble different LLMs' outputs for attribute value extraction. However, it is used only for two attributes (gender/age) and assumes all possible values are known beforehand. [19] is a contemporary work that uses LLMs along with retrieved examples from a fine-tuned T5 model to extract attribute value information. The usage of LLMs restricts the models from applications that need real-time inference, which we are able to solve by the use of our framework (Figure 1).

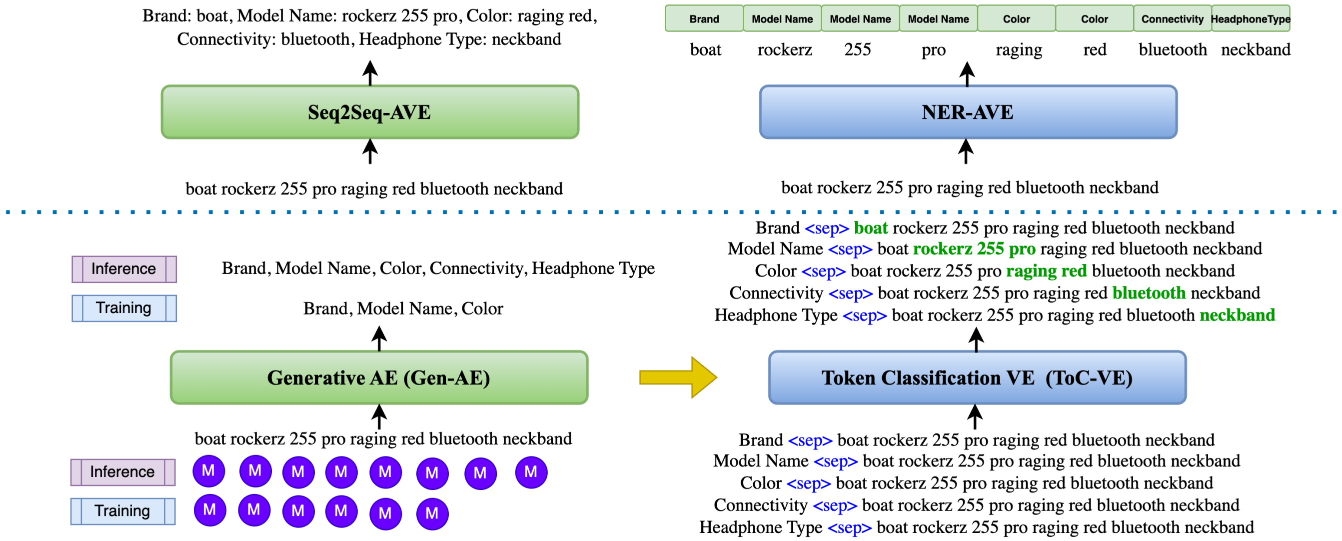

Figure 2: Model architectures. (a) Seq2Seq-AVE outputs a string that concatenates all attribute-value pairs for a given input query. (b) NER-AVE classifies each word in the query, tagging it with the relevant attribute. (c) GenToC employs Gen-AE to yield a concatenated list of attributes and ToC-VE to annotate the values linked to every recognized attribute. The Gen-AE model incorporates markers ('M') during the training process for the words which are covered. During inference, these markers are applied to all the words in the query.

Table 2: Comparison of different types of models.

Model	Partially labeled	Long-tail attributes	Response time
NER	✗	✗	fast
Seq2Seq	✗	✓	slow
GenToC	✓	✓	slow

3 Problem Statement

Given a query q with k words, where $q = w_1 w_2 \ldots w_k$, and a set of all potential attributes $\mathbb{A}$, the objective of attribute-value extraction is to identify all possible attribute-value pairs, denoted as $\{(a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n)\}$, where $a_i \in \mathbb{A}$ and v_i is a subset of words in q. In our setup, the input q may be a product title or user search query, with each word w in q linked to at most one attribute.

4 Background

In this section, we describe in detail two popular architectures that have been commonly used for the task of attribute-value extraction. We illustrate their working in Figure 2 (a) and (b).

4.1 NER-AVE

Following NER models used for the task [15, 21], NER-AVE is an encoder-only model that operates by classifying tokens in the input query, where each token in the query is assigned a label corresponding to the relevant attribute. For example, *boat* is assigned the attribute Brand, each of three words *rockerz*, *255* and *pro* are assigned the attribute Model Name, and so on. In case no attribute exists for a particular word, a special label, NoAttribute is used to indicate the same. As an NER model, it treats every attribute as a unique label and assigns an atomic embedding. However, its major strength lies in its speed due to the simple encoder-only architecture. Although it is not capable of effectively learning from partially labeled data, we find that training it with better quality data generated by GenToC, results in a fast and powerful attribute-value extraction system.

4.2 Seq2Seq-AVE

The Seq2Seq-AVE model is based on the model developed in Shinzato et al. [22], which employs a Seq2Seq model that yields a concatenated string, incorporating the respective attribute-value pairs, for a given input query. For instance, the encoder takes an input q, and the decoder generates the output string '$a_1{:}v_1, a_2{:}v_2, \ldots, a_n{:}v_n$'. For example, the decoder produces the generation "*Brand: boat, Model Name: rockerz 255 pro ...*", one token at a time. The attribute and value pairs can then be parsed from this generation. Its limitations are slow inference speed (found to be over 10x slower than NER-AVE) and limited learning ability from partially labeled data.

5 Methods

In this section, we describe the two-stage GenToC model and the marker-based training that enables it to learn from partially-labeled data. We show the working of GenToC in Figure 2 (c). The initial step employs a Generative Attribute Extraction (Gen-AE) model, which takes the product title or user search query as input and subsequently generates a concatenated list of predicted attributes. The model is trained to generate attributes in the order their values occur in the input. Given the model's generative ability, it might generate attributes outside the training set. However, these are very infrequent (occur less than 0.1% of the times), so we forgo constrained generation [3], avoiding any restrictions that would force the attributes to belong solely to the original set of attributes.

Markers: To deal with partially-labeled training data, we initially identify the words within the input query that correspond to values of any attribute. We refer to these identified words as *marked words*. For the example, "*Boat Rockerz 255 Pro Raging Red Bluetooth Neckband*", in Table 1, [*boat*, *rockerz*, *255*, *pro*, *raging*, *red*] are treated as marked words during training, as they have been labeled with some attribute. A special learnable embedding, termed as "*marker embedding*", is added to the encoder's final hidden states of every token of the marked words before being passed to the decoder. The same learnable embedding is shared across all marked tokens everywhere. The model is then able to learn that the output attributes, as observed in the training data, are a result of considering only the limited set of words in the input that have been marked. In other words, the marker embeddings act as signals that instruct the model to focus on and incorporate the information from these marked words into its attribute generation process, while preserving the broader context of the input query.

At inference time, the marker embedding is applied to all words within the input query, nudging the model to generalize and output attributes that are relevant to any word present in the query. This enhances the model's capability to recognize and associate attributes with words like *bluetooth* and *neckband*, which did not have valid attribute-value pairs annotated and hence were not marked at training time.

In the next stage, a Token Classification Value Extraction (ToC-VE) model takes each of the attribute names from the first stage along with the original query, separated by a special delimiter, *<sep>*. It then labels each token with a binary value, for *yes* or *no*, to denote whether it corresponds to a value or not for the chosen attribute. This model is trained independently of the first-stage model. Since attribute *names* are used in the training process, it allows the model to learn from closely related attributes which share similarities in their names. This becomes particularly crucial when dealing with a noisy attribute name ontology, which may include different attribute names conveying the same property, such as Model Number and Model No.

Value Pruning: When training the ToC-VE model with a set of attribute-value pairs, it always contains a value for every attribute. However, during the inference phase, the attributes may be erroneously generated by the Gen-AE model. As a consequence, ToC-VE model would tend to assign some value to even the incorrect attributes. To counteract this, we supplement the ToC-VE training with additional data to identify instances where no correct value is present for a given attribute. We accomplish this by taking an existing attribute-value pair and deleting the value from the original query. Consequently, we ensure that the chosen attribute no longer appears in the query, training the model to label all tokens as 'NO' values for the attribute. For example, if we remove the token, '*boat*', the example, "*brand <sep> rockerz 255 pro raging red bluetooth neckband*", will have no value for the attribute brand. We term the generation of this kind of synthetic training data as Value Pruning. For the final set of attribute-value pairs yielded by the model, we exclude those attributes that do not have any associated values.

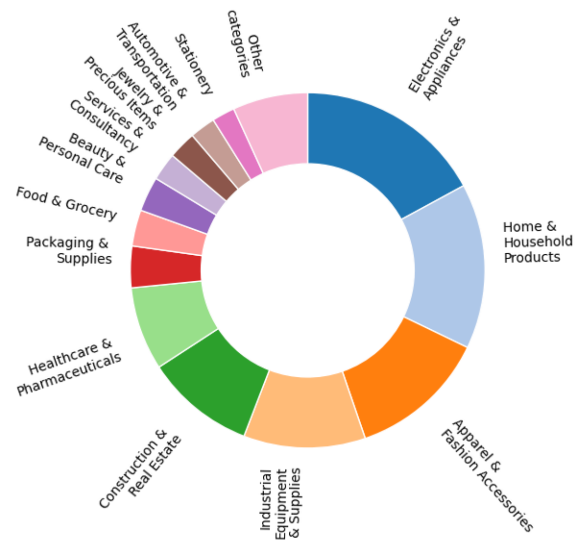

Figure 3: Distribution of product categories within training dataset. Specific percentage values are omitted to preserve data confidentiality.

Therefore, the architecture of GenToC ensures effective attribute-value pair extraction, preventing error build-up in the pipeline, even in the face of partially labeled data and vast attribute sets that contain redundancies. The complete algorithm for training and inference are described in Algorithm 1 and Algorithm 2 (in Appendix), respectively.

6 Experimental Setup

6.1 Dataset and Metrics

To curate data for the attribute-value extraction task, we make use of the product specifications that are provided by the sellers for product listings on IndiaMART. IndiaMART has 194M buyers, 7.9M sellers, and features over 108M different products and services. For the experiments in this paper, we limit our input exclusively to product titles and use a set of 2M examples for training models. It comprises 715K unique attribute-value pairs covering 24.7K attributes. The category-wise distribution of products is presented in Figure 3. Table 3 lists the top-five attributes and provides a set of example values for each. Despite the large scale of attributes, only 40.7% of the words of a product title are tagged on average with an attribute. This motivates us to build models designed for partially labeled data, where the present attribute-value pairs are reliable and of high quality but might lack the complete set of attribute-values.

We evaluate the systems on 39,671 samples (**39K**) based on the ground truth values available. We compute the precision, recall and F1-score by comparing the set of attribute-value pairs generated for each input with the ground truth set of attribute-value pairs. We then report their averages taken across all examples.

However, we find that automatic evaluation is challenging and often unreliable due to the lack of normalization in the attribute names, as the ground truth set can use different attributes to express

Table 3: Top-5 attributes in the dataset (based on frequency), along with a set of example values for each.

Attribute	Values
Brand	*hp, samsung, dell, siemens, bosch*
Material	*stainless steel, plastic, brass, wooden, cotton*
Color	*black, white, blue, red, brown*
Model name/number	*n95, classic, 12a, kn95, eco*
Usage	*industrial, office, kitchen, packaging, home*

the same characteristic. So, we randomly sample 2,000 examples (**2K**) and get every output checked using 3 data annotators (DAs). These annotators are skilled professionals who perform various annotation tasks within our organization. We consider the majority class assigned by the DAs to determine if a given attribute-value pair is correct/incorrect. We find that the inter-annotator agreement, computed using Fleiss' kappa [5], is $\kappa = 0.73$, indicating substantial agreement among the annotators [10]. For calculating recall, we pool all correctly marked attribute-value pairs from the various systems considered for evaluation. These pooled set of correct attribute-value pairs are considered as the ground truth set for calculating recall of a system and the fraction of these pairs generated by a system gives it's recall. We rely on this measure of recall as it is infeasible for a human to list all attribute-value pairs when there are thousands of possible attributes.

The response time is measured by calculating the average time taken for queries from the 2K test set (with a batch size of one) using an NVIDIA GeForce RTX 3090 GPU.

6.2 Implementation and Systems Compared

To compare GenToC with the two classes of state-of-art models, we use NER-AVE as a representative of the available NER-style models [15, 21, 30] and Seq2Seq-AVE as a representative of the published generative models [18, 22]. We use our own implementations due to the lack of availability of standard open-source implementations for the above works. For a fair comparison, we use the DeBERTa-V3-Small[2] [8] for all the classification tasks and BART-Base[3] [11] for all generation tasks. Both architectures consist of 6 encoder layers, and BART has 6 additional decoder layers. Due to the need for industry-scale systems to be used by millions of users, we do not experiment with LLMs, which are highly resource intensive. We release the code for training and inference at https://github.com/KnowDisAI/GenToC. Due to proprietary restrictions, the dataset utilized in our experiments is not available for release.

7 Experiments and Results

In this section, we answer the following questions:

(1) How does GenToC compare to other models using both automated and manual evaluation?
(2) What is the incremental contribution of Markers and Value Pruning in the GenToC system?
(3) Can GenToC improve training data tagging and thus be used to train faster models?
(4) What are the trade-offs between precision and recall in various models at different confidence thresholds?
(5) How is the system utilized and how does it perform in a production environment?

[2]https://hf.co/microsoft/deberta-v3-small
[3]https://hf.co/facebook/bart-base

7.1 Comparison with Other Models

In Table 4, we present a comparison of three systems – NER-AVE, Seq2Seq-AVE and GenToC. GenToC model outperforms its counterparts in recall, achieving 71.8% in automatic and 80.1% in manual evaluations, which are the highest among the three models. These scores are notably higher (by 11.7% and 27.3% in automatic and manual evaluations) than those of the next best-performing model, NER-AVE. Such a significant gap in recall demonstrates that GenToC effectively leverages the partially labeled nature of the training data, resulting in more extractions. Indeed, the total number of correct attribute-value extractions rose from 2,636 to 4,121 (a 56.3% increase) on the 2K test set. This increase in recall is mainly due to Markers, which is elaborated upon in Section 7.2. In terms of precision, both the NER-AVE and Seq2Seq-AVE models exhibit higher precision than the GenToC model in both automatic and manual evaluations, with approximately 9-10% and 4-5% higher precision, respectively. Overall, GenToC maintains its superiority in F1-score, recording the highest at 71.3% in automatic evaluations and 83.0% in manual evaluations, indicating a more balanced and consistent performance. However, a drawback of GenToC is its slower response time compared to the NER-AVE model. We address a potential solution to this performance-speed trade-off in Section 7.3.

In Table 6, we present example cases showcasing GenToC's superior performance in comparison to both NER-AVE and Seq2Seq-AVE models, in terms of the number of attribute-value pairs extracted.

NER-AVE and Seq2Seq-AVE models are unable to leverage the partially-labeled nature of the training data, as indicated previously in Table 2, despite possibly encountering the remaining unlabeled attribute-value pairs in other examples. In particular, each training example is tagged with only 1.53 attribute-value pairs on average, so it's not unexpected that these models maintain a similar tagging frequency during inference. GenToC addresses this challenge by incorporating Markers, which allow for the identification of more number of attribute names than what is typically found in the training data.

For instance, in the product title "*Casual Juliet Sleeve Solid Women Maroon Top*" from the first example in Table 6, the GenToC model surpasses the other models by identifying four additional attributes – Occasion, Sleeves Type, Pattern and Gender.

7.2 Ablation Study

Our initial hypothesis posited that incorporating Markers would enhance recall by identifying more attributes (critical in dealing with partially-labeled data), while Value Pruning (VP) would boost precision by eliminating inaccurately generated attributes. To validate this, we evaluated variations of the GenToC model, specifically configurations excluding either Markers, VP, or both. The outcomes, as presented in Table 4, align with our expectations. Notably, in the 2K test set, the absence of Markers in the GenToC model resulted in a 29.1% decrease in recall, while omitting VP led to a 7.2% reduction in precision. This pattern was also evident in the larger 39K test set.

Table 4: Model Performance. A comparison of NER-AVE, Seq2Seq-AVE and GenToC models reveals that GenToC performs superiorly in both automated and manual evaluations in terms of F1-score. However, it comes at the cost of increased response time with respect to NER-AVE and thus motivates our bootstrapping experiments.

Architecture	Automatic Evaluation (39K)			Manual Evaluation (2K)			Response Time (in ms/query)
	Precision	Recall	F1-score	Precision	Recall	F1-score	
NER-AVE	80.6	60.1	68.9	90.8	52.8	66.8	**8.8** ± 1.1
with Marker	58.6	71.2	64.3	66.5	**87.5**	75.6	**8.9** ± 1.1
Seq2Seq-AVE	80.1	52.5	63.4	89.6	50.7	64.8	91.9 ± 23.8
with Marker	60.0	67.9	63.7	65.2	69.9	67.5	149.4 ± 43.5
GenToC	70.8	71.8	**71.3**	86.1	80.1	**83.0**	90.1 ± 19.6
without Marker	**80.8**	54.7	65.2	**92.7**	51.0	65.8	66.1 ± 12.6
without VP	66.3	**74.3**	70.1	78.9	85.4	82.0	86.3 ± 18.8
without Marker & VP	79.7	55.1	65.2	90.7	52.2	66.3	66.5 ± 12.2

Table 5: Bootstrapping performance. GenToC and NER-AVE are re-trained with data bootstrapped from the two models. Results for GenToC and NER-AVE trained on original data are included for reference. NER-AVE trained with data bootstrapped from GenToC gives us the best trade-off between performance and speed for deployability.

Data Source	Architecture	Automatic Evaluation (39K)			Manual Evaluation (2K)			Response Time (in ms/query)
		Precision	Recall	F1-score	Precision	Recall	F1-score	
Original	NER-AVE	**80.6**	60.1	68.9	**90.8**	52.8	66.8	**8.8** ± 1.1
	GenToC	70.8	71.8	71.3	86.1	80.1	83.0	90.1 ± 19.6
NER-AVE	NER-AVE	80.1	61.3	69.5	90.0	55.3	68.5	**8.4** ± 1.1
	GenToC	70.7	68.8	69.7	86.8	76.8	81.5	94.2 ± 19.2
GenToC	NER-AVE	70.9	**72.4**	**71.6**	85.6	81.6	83.6	**8.4** ± 1.2
	GenToC	64.5	**72.4**	68.2	83.2	**91.2**	**87.0**	98.6 ± 21.6

Table 6: Predictions made by NER-AVE, Seq2Seq-AVE, and GenToC for three test examples. GenToC correctly identifies more attribute-value pairs in comparison to NER-AVE and Seq2Seq-AVE. (*SS* stands for *stainless steel*)

Product Title	NER-AVE	Seq2Seq-AVE	GenToC
Casual Juliet Sleeve Solid Women Maroon Top	Color: *Maroon*	Color: *Maroon*	Occasion: *Casual* Sleeves Type: *Juliet Sleeve* Pattern: *Solid* Gender: *Women* Color: *Maroon*
Sofar Ongrid Inverter 5.5KTL-X 5.5KW Three Phase	Brand: *Sofar*	Brand: *Sofar* Model: *5.5KTL-X*	Brand: *Sofar* Grid Type: *Ongrid* Model: *5.5KTL-X* Capacity: *5.5KW* Phase: *Three Phase*
Globe SS Induction Pressure Cooker	Brand: *Globe*	Brand: *Globe*	Brand: *Globe* Material: *SS* Type of Pressure Cookers: *Induction*

The findings clearly demonstrate the critical roles of both Markers and VP in balancing recall and precision. The GenToC model, when devoid of either component, shows a diminished F1-score compared to its complete configuration. These results substantiate our choice to incorporate both these strategies in the final formulation of the GenToC model, for its superior overall performance.

We also experiment with adding Markers to NER-AVE and Seq2Seq-AVE. During training, we incorporate marker embeddings to those words which have been covered by the available attributes, and at inference, we incorporate the marker embeddings to all words. Consistent with earlier practices, these embeddings are added into the final hidden states of the encoder. In the 2K test set, this modification results in considerable reductions in precision scores, plummeting to 66.5% from 90.8% in NER-AVE and to 65.2% from 89.6% in Seq2Seq-AVE. A similar pattern can be observed in case of the 39K test set as well. This outcome underscores the importance of two-stage model for using markers effectively. The bifurcated structure of the two-stage model ensures that markers are exclusively utilized in the first stage. This design circumvents the problem in single-stage models, where markers can force the assignment of attributes to every word, often resulting in suboptimal performance. This also significantly increases the response time for the Seq2Seq-AVE model (see Table 4), as it has to generate a longer output due to more extractions. On the other hand, Value Pruning is useful when there is an candidate set of attribute names that might include some incorrect ones, such as those generated by the first stage of the GenToC model. It aims to reduce irrelevant attribute names in the final result by predicting empty values for them. Conversely, Seq2Seq-AVE and NER-AVE produce both attribute-value pairs in a unified step, making Value Pruning inapplicable for them. Additionally, in Appendix B, we discuss the performance of the models with varying levels of partial-labeling in the training data.

7.3 Bootstrapping Training Data

From Table 4, we find that NER-AVE has the fastest response time. Seq2Seq-AVE is the slowest, as it has to generate a long string containing all attributes and values. The high performance of GenToC comes at the cost of speed, as its response time is 10x greater than NER-AVE's. With the motivation to build a deployable system that can handle real-world traffic, we experiment with cleaning the training data by regenerating it using the trained GenToC model. It increases the average number of words tagged in any attribute-value pair from 0.41 to 0.65 while increasing the total number of attribute-value pairs from 3M to 4.7M on the full training data. In Table 5, we find that the NER-AVE model trained with this GenToC-bootstrapped training data significantly improves the performance over the NER-AVE model trained with original partially-labeled data. In fact, its performance when trained with GenToC-bootstrapped data is comparable to the GenToC model in both automatic and manual evaluations. This makes it a strong alternative to the GenToC model for production environments, where rapid response times are crucial. Additionally, we experiment with regenerating the training data using the originally trained NER-AVE, the second-best model in terms of F1-score. However, we find that training with this data results in marginally decreased performance, proving less effective than training with GenToC-bootstrapped data. This is because, unlike GenToC, the NER-AVE model is not equipped to learn from incomplete data. Consequently, it does not help in reducing the partially labeled nature of the training data upon regeneration, rendering the bootstrapping unproductive.

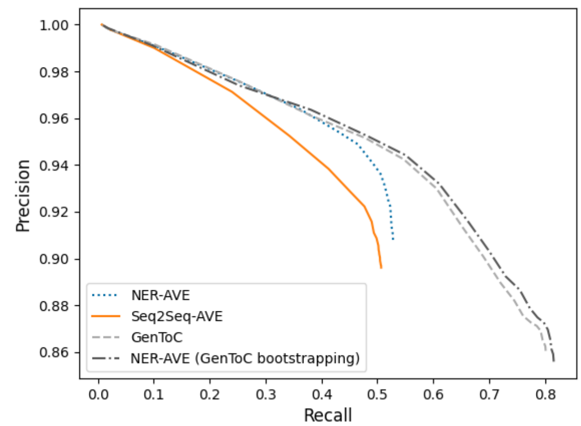

Figure 4: Precision-Recall curves show that GenToC and NER-AVE (GenToC bootstrapping) significantly outperform remaining models, NER-AVE and Seq2Seq-AVE.

In Appendix C, we present the performance of individual attribute and value extraction to provide a more comprehensive overview. Furthermore, in Appendices D and E, we also assess the performance of these models on long product titles and long-tail attributes. We find that GenToC outperforms Seq2Seq-AVE and NER-AVE systems by more than 13-18% in F1-score for long product titles and 7-19% in precision for rare attributes while helping boostrapped NER-AVE achieve similar levels.

7.4 Precision-Recall Trade-off

To understand the trade-off between precision and recall across different systems, we evaluate the systems using a precision-recall curve. We employ a common re-scoring model to compute the confidence level for all system extractions in this process. The model used for re-scoring is an independent Seq2Seq model, trained by using the *product title* as the input and the '*attribute: value*' string as the output, utilizing the same training data. A product title associated with N attribute-value pairs will form N training examples for the model. The trained model computes the probability of an attribute-value pair generated from any system by summing the log generation probablity of the tokens in the attribute-value string. We found that the model provides us with better-calibrated scores for an attribute-value pair generated by any of the systems. Using the results from the manually evaluated (2K) set, we applied uniformly spaced thresholds between 0 and 1 for the probablity values to create the Precision-Recall curve shown in Figure 4. Upon inspection, it was clear that GenToC's performance significantly eclipsed that of baselines. Across most of the curve, GenToC achieves a higher precision for a given recall, demonstrating its power. Further, the

Table 7: Offline evaluation of previously deployed model and proposed model.

System	Precision	Recall	F1-score
Rule-based (prior deployment)	**91.9**	70.9	80.0
Proposed method	89.5	**91.1**	**90.3**

AUC in case of NER-AVE and Seq2Seq-AVE are 0.51 and 0.48 respectively, while that of GenToC is 0.76. This substantiates the superior quality and efficacy of the GenToC system.

Additionally, the curve for NER-AVE trained with GenToC bootstrapped data closely mirrors that of GenToC, with an AUC of 0.77. This further demonstrates the effectiveness of training with data bootstrapping from GenToC.

7.5 Deployment Status and Impact

Our system has been deployed on India's largest B2B e-commerce platform, IndiaMART, with 194M buyers and 7.9M sellers. It has been successfully integrated into the core product search functionality and has already served over 200M requests since deployment. Specifically, it extracts attribute-value pairs from product titles in listings, and enables *dynamic feature highlighting* of product features according to user search queries. Products usually have a large number of specifications associated with them, in the form of attribute-value pairs. For example, an *"Apple iPhone"* might include attributes like Memory: *256 GB*, Color: *Black*, and Size: *6.1 inches*. Given the limited screen sizes, especially when users search on smartphones, it becomes important to prioritize the attribute-value pairs that should be displayed to the user. Identifying the attribute-value pairs implied from user search queries such as *"256 GB iPhone"*, allows the highlighting of relevant attribute-value pairs that can be most informative to the user during their search journey.

For the deployment version, we train GenToC using a dataset that is 3x larger, with a similar partially-labeled nature. Further, we regenerate the training dataset using the trained GenToC model and subsequently train a faster NER-AVE model with this bootstrapped data, as detailed in Section 7.3. We use this GenToC-bootstrapped NER-AVE model for deployment.

We find significant improvements over the prior deployed system (which is based on rule-based non-neural system) for this task. The rule-based system utilizes a dictionary mapping of values to attributes based on the frequency of their occurrence in the existing product catalog. Regexes are used to identify spans in the product name mapped to any attributes in the dictionary. All such identified attribute-value pairs are reported by the system. Such regex-based systems have significant limitations, such as struggling with negations (e.g., *non-stretchable jeans*), handling common spelling variations (*litre*, *liter*, *l*, or *ltr*), making broad generalizations (both *red* and *raging red* refer to colors), and properly tagging attributes contextually (*Galaxy* could refer to either a chocolate brand or a Samsung mobile model).

In an offline evaluation, based on a manual audit of search queries, we observe the results tabulated in Table 7. It can be observed that the previously deployed rule-based system falls short of our proposed approach by more than 20% in terms of recall, while being only marginally better by less than 2.5% in precision. This results in an overall difference of approximately 10% in the F1-score between the two methods. In online evaluation, we find that our system leads to a 9% increase in queries with identified attributes that lead to dynamic feature highlighting.

8 Conclusion and Future Work

In this work, we introduce a new framework designed to effectively utilize incomplete training data, which is common in attribute-value extraction tasks, and provide a solution suitable for real-time deployment. We achieve this by employing the novel GenToC model for attribute-value extraction, incorporating *Markers* to learn from partially labeled data and *Value Pruning* to avoid erroneous attribute tagging. Moreover, by utilizing GenToC's ability to enhance training data through bootstrapping, we can train faster NER models that are not designed to learn from partially labeled data, thereby bridging the gap between research and practical application. Our system has been deployed on India's largest B2B e-commerce platform, IndiaMART, into the core product search functionality. Future work could explore performing multiple stages of bootstrapping, handling near-redundancy in attributes, and uncovering novel attributes.

Acknowledgments

We would like to thank IndiaMART for providing us with the data for our research and helping us integrate with their current systems. We would like to thank KnowDis Data Annotator team for their timely help with the annotations required for this project and KnowDis Data Science members for giving valuable suggestions. Mausam is supported by a consultancy grant from KnowDis and a Jai Gupta Chair fellowship.

References

[1] Lidong Bing, Tak-Lam Wong, and Wai Lam. 2012. Unsupervised extraction of popular product attributes from web sites. In *Information Retrieval Technology*. Springer Berlin Heidelberg, 437–446.

[2] Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2023. ExtractGPT: Exploring the Potential of Large Language Models for Product Attribute Value Extraction. In *International Conference on Information Integration and Web-based Applications & Services*.

[3] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=5k8F6UU39V

[4] Chenhao Fang, Xiaohan Li, Zezhong Fan, Jianpeng Xu, Kaushiki Nag, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2024. Llm-ensemble: Optimal large language model ensemble method for e-commerce product attribute value extraction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[5] Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76 (1971), 378–382.

[6] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew E. Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explor.* 8 (2006), 41–48.

[7] Vishrawas Gopalakrishnan, Suresh Iyengar, Amit Madaan, Rajeev Rastogi, and Srinivasan H. Sengamedu. 2012. Matching product titles using web-based enrichment. *Proceedings of the 21st ACM international conference on Information and knowledge management* (2012).

[8] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION. In *International Conference on Learning Representations*. https://openreview.net/forum?id=XPZIaotutsD

[9] Anant Khandelwal, Happy Mittal, Shreyas Kulkarni, and Deepak Gupta. 2023. Large Scale Generative Multimodal Attribute Extraction for E-commerce Attributes. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. Association for Computational

Linguistics, Toronto, Canada.

[10] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. http://www.jstor.org/stable/2529310

[11] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online.

[12] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30 (2007), 3–26.

[13] Katharina Probst, Rayid Ghani, Marko Krema, Andrew E Fano, and Yan Liu. 2007. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2838–2843.

[14] Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1557–1567.

[15] Martín Rezk, Laura Alonso Alemany, Lasguido Nio, and Ted Zhang. 2019. Accurate Product Attribute Extraction on the Field. *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), 1862–1873.

[16] Thomas Ricatte and Donato Crisostomi. 2023. AVEN-GR: Attribute Value Extraction and Normalization using product GRaphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams (Eds.). Association for Computational Linguistics, Toronto, Canada.

[17] Kalyani Roy, Pawan Goyal, and Manish Pandey. 2021. Attribute Value Generation from Product Title using Language Models. *Proceedings of The 4th Workshop on e-Commerce and NLP* (2021).

[18] Kalyani Roy, Pawan Goyal, and Manish Pandey. 2024. Exploring generative frameworks for product attribute value extraction. *Expert Systems with Applications* 243 (2024), 122850. https://doi.org/10.1016/j.eswa.2023.122850

[19] Kassem Sabeh, Mouna Kacimi, Johann Gamper, Robert Litschko, and Barbara Plank. 2024. Exploring Large Language Models for Product Attribute Value Identification. *ArXiv* abs/2409.12695 (2024).

[20] Keiji Shinzato and Satoshi Sekine. 2013. Unsupervised extraction of attributes and their values from product description. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, 1339–1347.

[21] Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022. Simple and Effective Knowledge-Driven Query Expansion for QA-Based Product Attribute Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Dublin, Ireland.

[22] Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2023. A Unified Generative Approach to Product Attribute-Value Identification. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada. https://aclanthology.org/2023.findings-acl.413/

[23] Anubhav Shrimal, Avi Jain, Kartik Mehta, and Promod Yenigalla. 2022. NER-MQMRC: Formulating Named Entity Recognition as Multi Question Machine Reading Comprehension. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*. Association for Computational Linguistics, Hybrid: Seattle, Washington + Online, 230–238. https://doi.org/10.18653/v1/2022.naacl-industry.26

[24] K. Wang, Jianzhi Shao, T. Zhang, Qijin Chen, and Chengfu Huo. 2023. MPKGAC: Multimodal Product Attribute Completion in E-commerce. *Companion Proceedings of the ACM Web Conference 2023* (2023).

[25] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit K. Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jonathan L. Elsas. 2020. Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).

[26] Yuk Wah Wong, Dominic Widdows, Tom Lokovic, and Kamal Nigam. 2009. Scalable Attribute-Value Extraction from Semi-structured Text. *2009 IEEE International Conference on Data Mining Workshops* (2009), 302–307.

[27] Huimin Xu, Wenting Wang, Xin Mao, Xinyue Jiang, and Man Lan. 2019. Scaling up Open Tagging from Tens to Thousands: Comprehension Empowered Attribute Value Extraction from Product Title. In *Association for Computational Linguistics (ACL)*.

[28] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit K. Sanghai, Bin Shu, Jonathan L. Elsas, and Bhargav Kanagal. 2021. MAVE: A Product Dataset for Multi-source Attribute Value Extraction. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (2021).

[29] Danqing Zhang, Zheng Li, Tianyu Cao, Chen Luo, Tony Wu, Hanqing Lu, Yiwei Song, Bing Yin, Tuo Zhao, and Qiang Yang. 2021. QUEACO: Borrowing Treasures from Weakly-labeled Behavior Data for Query Attribute Value Extraction. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).

[30] Guineng Zheng, Subhabrata Mukherjee, Xin Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018).

[31] Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product. In *Conference on Empirical Methods in Natural Language Processing*.

A GenToC Algorithms

Algorithm 1: Training algorithm for GenToC

Input: Set of <Query, Incomplete attribute-value pairs>
Output: Gen-AE and ToC-VE models

Training Gen-AE Model:
foreach *<query, attribute-value pairs> in training set* **do**
    Step-1: Tag the words as *marked* in the query if they are present in any of the attribute values.
    Step-2: Train Gen-AE with marker embeddings added to the encoder's hidden states of the marked words.
    Step-3: Gen-AE is trained to generate the attribute list in order of occurrence in the query.

Training ToC-VE Model:
foreach *each query in training set* **do**
    foreach *each attribute-value pair associated with the query* **do**
        Step 1: Concatenate both the attribute name and query with *<sep>* delimiter.
        Step 2: Label tokens as 'YES' if they are present in the value for the attribute; otherwise 'NO'.
        Step 3: Apply Value Pruning to create examples with no values for a particular attribute.
        Step 4: Train ToC-VE model for binary classification of token values ('YES'/'NO').

Algorithm 2: Inference algorithm for GenToC

Input: query
Output: List of attribute-value pairs

Step 1: Tag all words in the query as *marked*.
Step 2: Use the trained Gen-AE model to predict attributes for the query with marker embeddings added to encoder hidden states of the marked words.
Step 3: For each predicted attribute from Gen-AE: Step 3.1: Concatenate the predicted attribute and query with *<sep>* delimiter.
    Step 3.2: Use the trained ToC-VE model to classify each token as a value ('YES') or not ('NO').
    Step 3.3: Extract values for each attribute, excluding attributes with no associated values.

Table 8: Performance of NER-AVE and GenToC when trained at different levels of partial-labeling

Words Tagged in Train Data	# Training Examples	Architecture	Automatic Evaluation (39K)			Manual Evaluation (2K)		
			Precision	Recall	F1-score	Precision	Recall	F1-score
40% (Original)	2M	NER-AVE	80.6	60.1	68.9	90.8	60.6	72.7
		GenToC	70.8	71.8	71.3	86.1	89.1	87.6
50%	1.2M	NER-AVE	79.9	64.8	71.6	89.5	66.2	76.1
		GenToC	71.0	72.3	71.7	84.6	89.5	87.0
60%	634K	NER-AVE	79.3	71.5	75.2	86.6	70.5	77.7
		GenToC	71.2	72.8	72.0	83.2	88.4	85.8

Table 9: Individual performance of attribute and value extraction (automatic evaluation on 39K dataset)

Architecture	Attribute Extraction			Value Extraction		
	Precision	Recall	F1-score	Precision	Recall	F1-score
NER-AVE	**89.4**	68.5	77.6	85.2	63.4	72.7
with Marker	73.4	**89.7**	80.7	65.1	78.9	71.4
Seq2Seq-AVE	86.4	57.8	69.2	**86.7**	57.0	68.8
with Marker	75.9	84.0	79.8	66.4	75.1	70.5
GenToC	79.5	82.0	80.7	78.6	79.1	78.9
without Marker	87.7	60.6	71.6	**86.7**	58.7	70.0
without VP	75.2	85.1	79.8	75.5	**82.8**	79.0
without Marker & VP	86.9	61.2	71.8	86.0	59.4	70.2
NER-AVE (GenToC bootstrapping)	79.7	82.5	**81.1**	78.3	79.8	**79.1**

B Performance at Different Levels of Partial-Labeling

In this section, we investigate the behavior of the models at different levels of partial-labeling in the training dataset by artificially modifying it. As mentioned in Section 6.1, the original proportion of tagged words in the training dataset is around 40%. To increase the coverage, we randomly dropped inputs with a fraction of tagged words below a specific threshold, with a certain probability. These threshold and probability values were adjusted to achieve coverage levels of around 50% and 60%. Specifically, to reach 50% coverage, we dropped rows with less than 35% tagging with a probability of 0.75, while for 60% coverage, we dropped rows with less than 55% tagging with a probability of 0.9. This resulted in 1.2M training examples for 50% coverage and 634K examples for 60% coverage.

We evaluate our GenToC model, which is specifically designed to handle partially-labeled training data, and compare it with NER-AVE on the same test sets. The results are presented in Table 8. We observe that in the automatic evaluation, the F1-score gap between GenToC and NER-AVE models narrows as more words are labeled in the training data. Since automatic evaluation has its own limitations (as discussed in Section 6.1), we also perform a manual evaluation. Note that for recall computation in the manual evaluation, we pool predictions only from the models included in this specific experiment. A similar trend is observed in the manual evaluation as well: GenToC outperforms NER-AVE, with the F1-score gap widening as coverage decreases from 60% to 40%. This highlights GenToC's effectiveness in handling partially-labeled training data compared to NER-AVE.

C Individual Performance Analysis of Attribute and Value Extraction

In Section 7 (Tables 4 and 5), we presented the performance metrics of the models on the attribute-value extraction task, evaluated as unified pairs. In this section, we present a more granular view by separating the results for *Attribute Extraction* and *Value Extraction.* The automatic evaluation performance on the 39K dataset is tabulated in Table 9. We observe a significant improvement in recall when Markers, which effectively handle partially-labeled data, are included in the models. NER-AVE and Seq2Seq-AVE achieve the highest precision in attribute extraction and value extraction, respectively. Overall, NER-AVE bootstrapped with GenToC data outperforms other models in terms of F1-score for both attribute extraction and value extraction, followed by GenToC, when compared to NER-AVE and Seq2Seq-AVE. These trends in individual evaluation align with our results in unified attribute-value prediction in Sections 7.1 to 7.3.

Additionally, while GenToC is designed to function as a single unified system, the second stage of GenToC, namely the ToC-VE model, can also be used independently for identifying values when the set of attributes for an input is already known or the set of all possible attributes is small. We observe that the accuracy for the task of identifying the value given the correct attribute is 82.2% on the 39K test set.

D Performance on Long Product Titles

To assess how well the models handle long-tail cases, we evaluate their performance on long product titles. Product titles consist of five words on average, with a standard deviation of two. Thus, we create a test set comprising 500 examples, each containing at least seven words, for the evaluation process. In Table 10, we tabulate the scores obtained from manual evaluation, together with the *tagged ratio* — that is, the proportion of words in the product title that have been marked with an attribute — for various models.

While Seq2Seq-AVE exhibits the highest precision at 95.3%, and GenToC without VP shows a notable recall of 90.3%, GenToC, stands out with the highest F1-score of 90.7%. This underscores GenToC's superior balance in the precision-recall trade-off, even in cases where the product titles are long. It's noteworthy that when Markers are incorporated into the NER-AVE and Seq2Seq-AVE models, over 99% of the words in product titles are linked to an attribute. However, this high tagging ratio might not be ideal, particularly for lengthy queries, as not every word necessarily possesses a relevant attribute. This is also reflected in the significant decrease in precision values for these models when Markers are used — a drop of over 22% compared to their counterparts without Markers. Single-stage models, which have to perform value labeling alongside attribute extraction, face this issue when Markers are employed, as they attempt to arbitrarily assign an attribute to every word. However, since we apply markers only to the first stage of the GenToC model, it does not face this problem as there is no direct one-to-one mapping between the predicted attributes and input words. Finally, even the NER-AVE model trained with GenToC bootstrapping attains an F1-score that's comparable with GenToC, while having a faster response time. This shows that the full potential of NER-AVE model is realized only with high-quality training data.

Table 10: Results of manual evaluation for various models on a test set of 500 long product titles.

Architecture	Precision	Recall	F1-score	Tagged Ratio
NER-AVE	94.8	65.2	77.3	0.464
with Marker	72.4	90.3	80.4	**0.999**
Seq2Seq-AVE	**95.3**	58.2	72.2	0.403
with Marker	71.7	71.6	71.6	0.993
GenToC	90.7	90.7	**90.7**	0.662
without Marker	94.5	58.7	72.4	0.405
without VP	87.1	**93.0**	89.9	0.697
without Marker & VP	94.2	59.0	72.6	0.406
NER-AVE (GenToC bootstrapping)	90.5	88.4	89.4	0.689

E Performance on Long-tail Attributes

To assess the efficacy of models in handling attributes with low occurrence rates, we measure their precision on attributes within the bottom 33% by frequency in the training dataset. These findings are illustrated in Figure 5. Notably, the NER-AVE model, when trained on the original dataset, exhibits a markedly inferior performance, with a difference of up to 19% compared to the GenToC model with the bottom-10% of attributes. This is likely due to the NER-AVE model treating each attribute name as an independent atomic label, which, combined with the limited training data for long-tail attributes, makes learning difficult during training. Conversely, Seq2Seq-AVE and GenToC employ compositional encoding for attributes, enabling them to capture the semantic meaning of attribute names by considering the constituent words. This approach allows them to handle long-tail and complex attributes more effectively than NER-AVE. Additionally, it can be seen that NER-AVE with GenToC bootstrapping performs better than when trained with the original data. This improvement is likely due to the enrichment of attribute names provided by the GenToC model during the bootstrapping process.

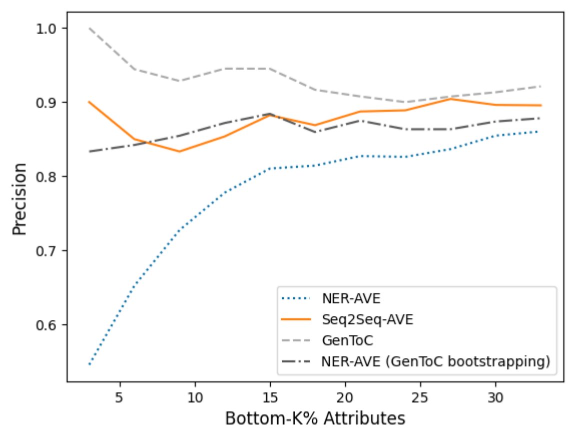

Figure 5: Performance of NER-AVE, Seq2Seq-AVE, GenToC, and GenToC-bootstrapped NER-AVE on long-tail attribute names. NER-AVE trained on original data shows poor performance on infrequent attribute names.